Mask Set Errata 2

68HC908AZ60 8-Bit Microcontroller Unit

INTRODUCTION

This mask set errata provides information pertaining to the MSCAN, SPI, and TIM modules and the flash memory programming time applicable to this 68HC908AZ60 MCU mask set device:

- 3H62A

MCU DEVICE MASK SET IDENTIFICATION

The mask set is identified by a 5-character code consisting of a version number, a letter, two numerical digits, and a letter, for example 3H62A. Slight variations to the mask set identification code may result in an altered version number, for example 4H62A.

MCU DEVICE DATE CODES

Device markings indicate the week of manufacture and the mask set used. The date is coded as four numerical digits where the first two digits indicate the year and the last two digits indicate the work week. For instance, the date code “9115” indicates the 15th week of the year 1991.

MCU DEVICE PART NUMBER PREFIXES

Some MCU samples and devices are marked with an ZC or XC prefix. A ZC prefix denotes special/custom device. An XC prefix denotes that the device is tested but is not fully characterized or qualified over the full range of normal manufacturing process variations. After full characterization and qualification, devices will be marked with the MC prefix.

When contacting a Motorola representative for assistance, please have the MCU device mask set and date code information available.

MSE Published Date: April 27, 1998

MSCAN RECEIVE AND TRANSMIT ERROR COUNTERS

Correct operation of the MSCAN Receive and Transmit Error Counters cannot be assured when running at high temperatures across all voltages. The counters can decrement by a count of 1, 2, 3, or 4 due to weak P-channel devices within the error counter control logic.

This problem arises when one to three errors have occurred in the system and after receipt or transmission of a successful message. The error counter decrements by four instead of one. This immediately puts the MSCAN module into Receiver Error Passive or Transmitter Error Passive mode. Normal operation will resume once the error counter returns to below the limit of 127.

Special tests have been developed for the 3H62A mask set to screen for this failure mode. However, samples are very limited due to excessive yield loss. Standard processing does not output to screen for this failure.

MSCAN RECEIVE MESSAGE CORRUPTION

There exists the possibility that under a particular set of circumstances, the msCAN receive buffer will contain a corrupted message. Under normal operation, it is expected that the required circumstances for the error will occur infrequently.

The msCAN background receive buffer will contain a corrupt message after either of the following sequence of events have occurred:

A. Overrun Condition

A.1

Initial state: Both receive buffers (foreground and background) are filled by accepted messages. They have not yet been released by the software.

A.2

Another message is received, triggering the overrun condition (OVRIF = 1).

A.3

The software releases one (or both) of the receive buffers by clearing RXF once (or twice) during the reception of a message (either the message which triggers the overrun condition or a subsequent message).

B. “Almost” Overrun Condition

B.1

Initial state: Both receive buffers (foreground and background) are filled by accepted messages. They have not yet been released by the software.

B.2

The msCAN begins to transmit a message M.

B.3

Software releases one (or both) of the receive buffers by clearing RXF once (or twice) during the transmission of the arbitration field of message M.

B.4

The msCAN loses arbitration on the CAN bus while transmitting M.

If either of the above sequences occur, the next bits seen on the bus after clearing RxF will be interpreted as the beginning of the ID field and will be transferred into the background receive buffer. The background buffer will then contain a shifted image of the true message. This message will be brought into the foreground buffer when the foreground buffer is next released ONLY if the shifted identifier happens to pass the acceptance filter configuration programmed into the msCAN.

The erratum is caused by a functional error in the msCAN design implementation.

To avoid the error conditions the msCAN receive driver software must process the incoming stream of messages fast enough to prevent both receive buffers becoming filled with accepted messages. This may be achieved by optimizing the interrupt service routines with higher priority than the msCAN receive interrupt. Alternatively, the msCAN receive and transmit driver software may be modified to detect the circumstances under which the error occurs.

SERIAL PERIPHERAL INTERFACE (SPI)

Clearing the SPE bit to disable the SPI can cause an error when transmitting in slave mode. In this situation, a race condition occurs, allowing an invalid mode fault to occur.

Mode faults occur on the SPI when the slave select ($\overline{SS}$) pin is toggled during a transmission. Mode faults also occur if $\overline{SS}$ is selected and then unselected before SPSCK returns to its idle level after the shift of the eighth data bit when CPHA = 0 while in slave mode.

When the SPI is disabled, the special port function associated with $\overline{SS}$ is also disabled and returns to a logic 1. In slave mode, $\overline{SS}$ must remain a logic 0 during a transmission. Thus, disabling the SPI causes the $\overline{SS}$ signal to go high internally, which sets up a race for the port logic to send in a logic 1 and the SPI to shut down mode fault detection internally.

This condition can be avoided easily in software if mode faults are disabled by clearing the MODFEN bit of the SPSCR register before disabling the SPI in slave mode.

When the toggle on overflow (TOV) bit is set, writing to a TCHxH register at the point of an overflow inhibits the associated pin from toggling until the TCHxL register is written. The pin then toggles at the next overflow. Even though a toggle can be completely missed, the TOF flag will be set and an interrupt can be generated. The only way to inhibit a toggle on overflow and set the TOF bit is to write to the TMODH register until the TMODL register is written. Similarly, in buffered PWM mode, writing to the inactive registers (TCH0H:l, TCH2H:L, TCH4H:L) at this overflow point produces the same problem. Writing to the odd channels (TCH1H:L, TCH3H:L, TCH5H:L) produces no faults.

Avoid this problem by using the overflow routine instead of writing to inactive channel registers within the output compare routine. Each output compare event occurs as a result of the last channel register written to prior to the last overflow.

Make sure that both odd and even timer channel registers are initialized. Write to the odd channels last, because the active channel register on startup is the even channel. If the inactive channel register is not written to last, then the next PWM pulse width will be exactly the same as the first, reflecting the value written to the even channel register.

FLASH MEMORY PROGRAMMING TIME

The minimum flash bus clock period (t_{CYC}) is 119 ns (8.4MHz).

The minimum flash erase time (t_{ERASE}) is 100 ms. The maximum is 110 ms.

Verify mode is now known as margin read mode.

When using the iterative program/margin read technique, a maximum of five programming pulses is allowed. The nominal duration of the programming pulse (t_{STEP}) is 1 ms. The t_{STEP} duration is defined as the amount of time during one program cycle that HVEN is held high. The minimum t_{STEP} is 0.8 ms and the maximum t_{STEP} is 1.2 ms. Therefore, the maximum cumulative program time (t_{PROG}) per page is 6 ms.

The minimum program time (t_{PROG}) is 0.8 ms. The maximum program time is 6 ms.

Cumulative program time exceeding 6 ms may cause unintentional programming of erased bits. This condition is known as program disturb.

The maximum program time per row between erase cycles is 48 ms. Program time greater than 48 ms per row may cause program disturb.

NOTE: *The above timings apply to devices which have been through a production test flow including burn-in. For development samples (marked PC) this is not always possible and in these cases a maximum of twenty program/verify iterations should be used to program the Flash.*

The minimum flash endurance is 100 erase/program cycles.

The minimum flash block endurance is 100 cycles.

FLASH MEMORY MAXIMUM FREQUENCY

Characterization of the flash memory across the full temperature/voltage/frequency specification has detected a fault which prevents reads of the flash memory at high temperatures. The maximum frequency versus temperature is defined as:

Temperature	Maximum Frequency
– 40 °C	8.4 MHz
25 °C	8.4 MHz
85 ∞∞°C	6.0 MHz
125 °C	5.0 MHz

LOW CURRENT MODES EXECUTED FROM FLASH

Executing STOP and WAIT instructions from code running in the Flash may cause high STOP and WAIT IDD. There are two possible software work-arounds to invoke STOP (WAIT) mode with minimum IDD consumption. One technique involves jumping to RAM to execute the STOP (WAIT) instruction. Another technique entails placing the STOP (WAIT) instruction at the end of a Flash memory boundary where the next location is not a Flash location. For example, placing the STOP (WAIT) instruction at $FDFF, since the next location $FE00, is not a Flash address.

Additional mask set erratas can be found on the World Wide Web at http://Design-NET.com/csic/TECHSPRT/TechSprt.htm.